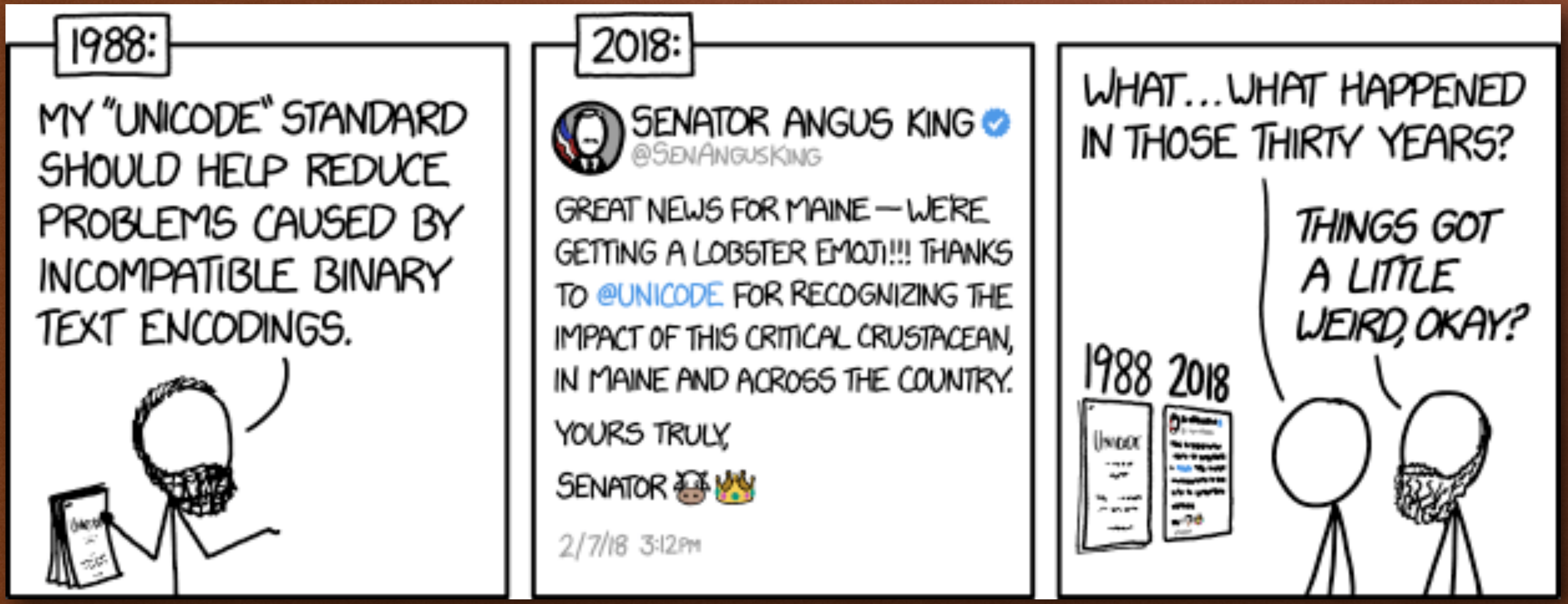
THE BUILDING BLOCKS:
ENCODING OTHER DATA



https://xkcd.com/1953/

LEARNING OBJECTIVES

- How do we encode text
    - ASCII and Unicode (UTF-8)
- Text compression
    - Huffman encoding
- Run-length encoding
- Representing sounds and images
    - Lossy compression

UTF-8

- The most commonly used Unicode character set encoding. Unicode Transformation Format 8-bit.
- Variable width - 1 to 4 bytes
- 1,114,112 "code points" numerical values corresponding to characters or formatting
- More frequently occurring code points are encoded in fewer bytes.

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

EMOJIS HAVE UTF-8 ENCODINGS TOO

- 📱
  - mobile phone
  - Unicode: U+1F4F1, UTF-8: F0 9F 93 B1
- 🇳🇿
  - flag of New Zealand
  - Unicode: U+1F1F3 U+1F1FF, UTF-8: F0 9F 87 B3 F0 9F 87 BF
    - this uses 8 bytes, it is a supplementary character (don’t worry)

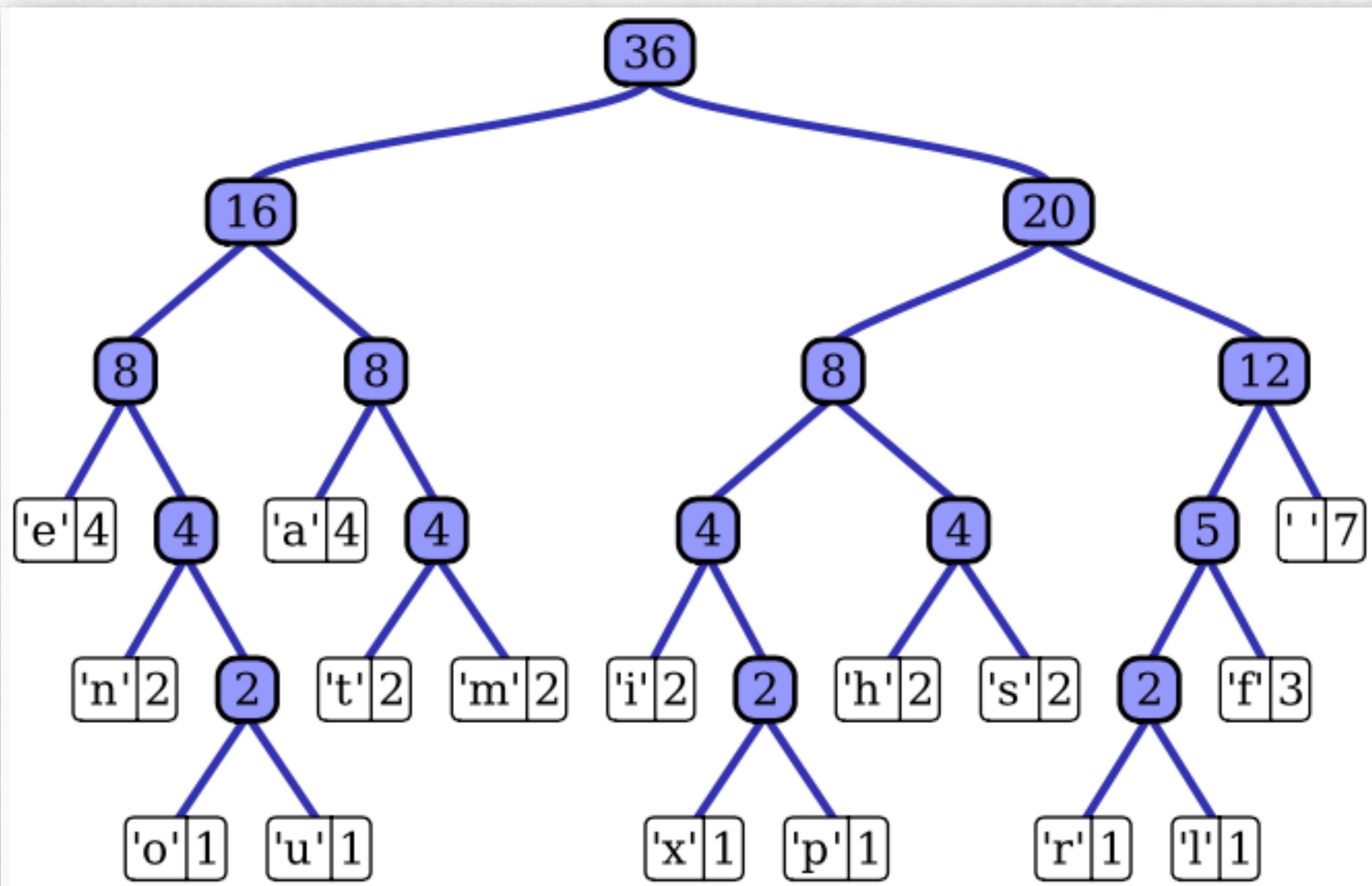
COMPRESSION

- As we just saw UTF-8 doesn’t need to use 4 bytes for each character. If the characters are ASCII they only take up 1 byte each.
- We can try to do better with the smallest encoding of the most common characters - this is the basis of Huffman coding.
- https://www.youtube.com/watch?v=JsTptu56GM8

HUFFMAN TREE

- This tree from Wikipedia is the Huffman tree for the text "this is an example of a huffman tree".

- It can also be represented as a table



Letter	Code
e	000
a	010
space	111
n	0010
t	0110
m	0111
i	1000
h	1010
s	1011
f	1101
o	00110
u	00111
x	10010
p	10011
r	11000
l	11001

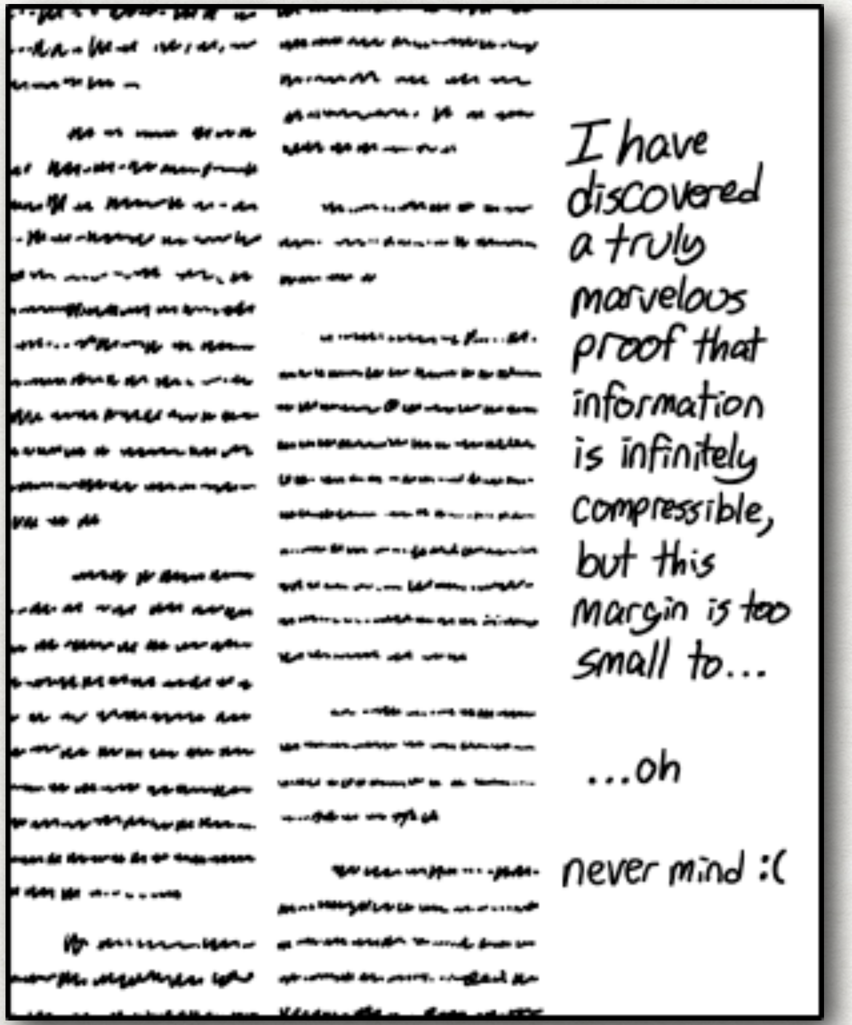
COMPRESSION RATIO

- How good is our compression?
- The compression ratio = $\frac{\text{uncompressed size}}{\text{compressed size}}$
- With the example on the previous slide the number of bits to represent the message is 195 and the uncompressed ASCII equivalent (one char per byte) is 36 × 8 = 288. *This assumes the tree or table doesn't have to be transmitted.*
- So the compression ratio is 288/195 ≅ 1.48
- But if we only had the 16 characters in the message to encode in a fixed length encoding we could do that with 4 bits per character.
- This would give a compression ratio of (36 x 4) / 195 = 144/195 = 0.74 Aargh!!! worse

Something to do - check the value of 195 above.

COMPRESSION WHY BOTHER?

- With so much memory available today, why do we bother with compression?



https://xkcd.com/1381/

RUN LENGTH ENCODING (RLE)

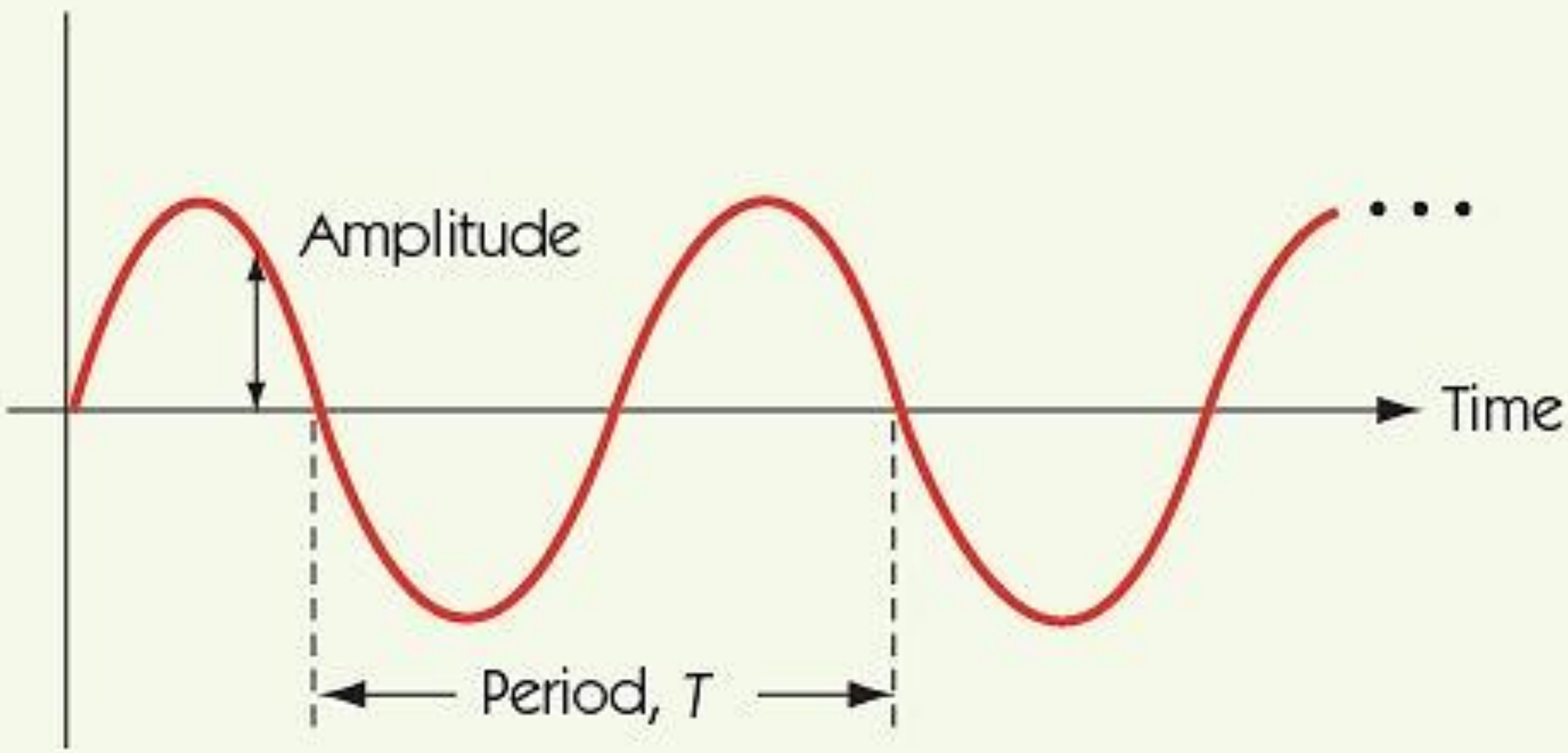
- A totally different form of compression.
- Not really useful for text, but for certain types of graphics.
- Values repeated in a row are counted and we store the value and the count
  - e.g. AAABBCCCC could be represented as A3B2C4
    - a compression ratio of 9/6 = 1.5
- Does RLE help with the sentence "this is an example of a huffman tree"?
- Other forms of compression look at groupings of characters (this can give much better compression ratios) https://en.wikipedia.org/wiki/Lempel–Ziv–Welch

SOUNDS AND IMAGES

- Sounds and images require converting naturally **analogue representations** to **digital representations**
- Sound waves characterized by:
  - **Amplitude:** height of the wave at a moment in time
  - **Period:** length of time until wave pattern repeats
  - **Frequency:** number of cycles per unit time

SOUND WAVE

FIGURE 4.4



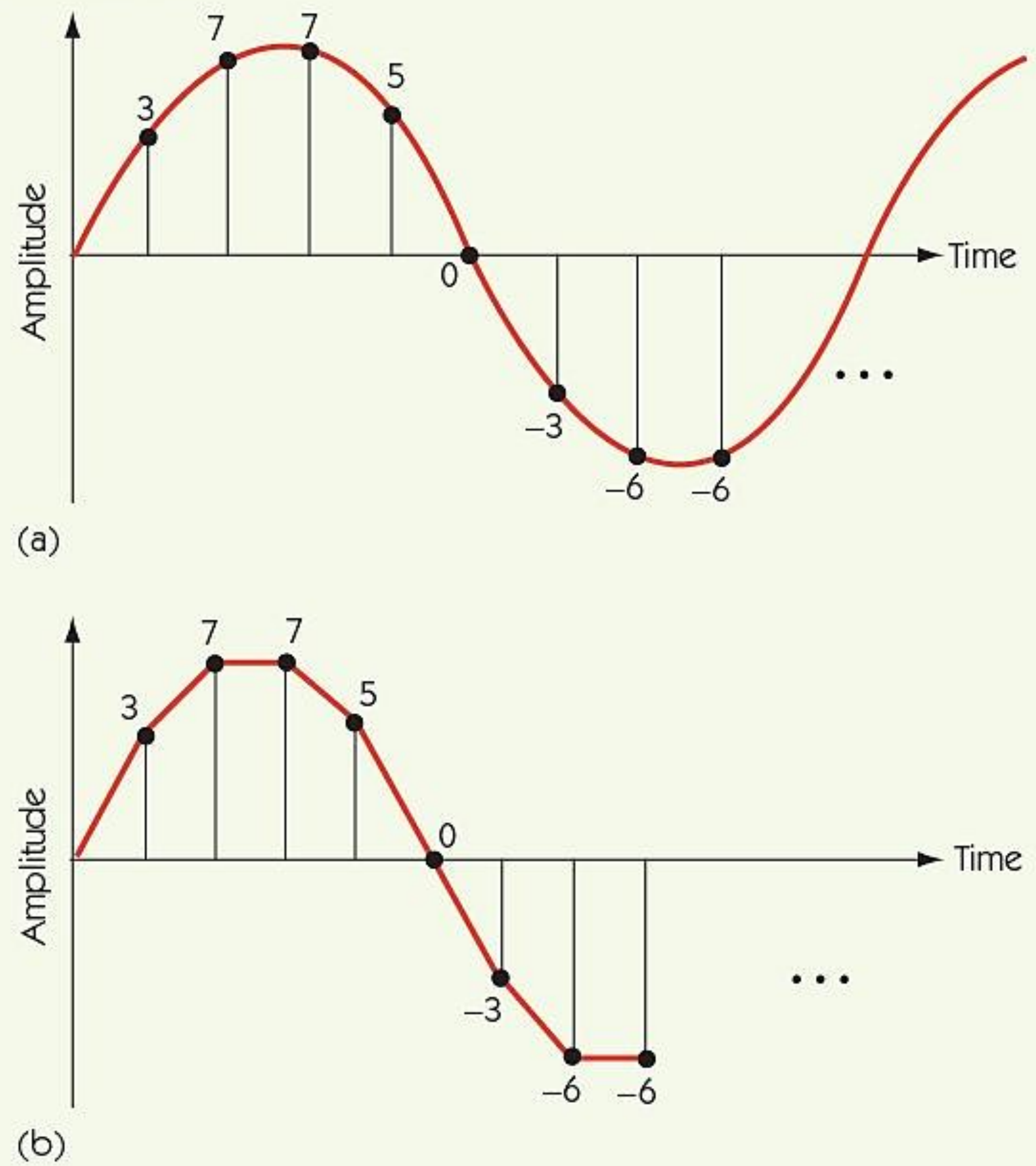
Example of sound represented as a waveform

ANALOGUE TO DIGITAL

- **Digitize**: to convert to a digital form
- **Sampling**: record sound wave values at fixed, discrete intervals
- To reproduce sound, approximate using samples
- Quality is determined by
  - **Sampling rate:** number of samples per second
    - More samples ▶ more accurate waveform
  - **Bit depth:** number of bits per sample
    - More bits ▶ more accurate amplitude

SAMPLING

FIGURE 4.5



Digitization of an analog signal
(a) Sampling the original signal
(b) Recreating the signal from the sampled values

TOO MUCH DATA

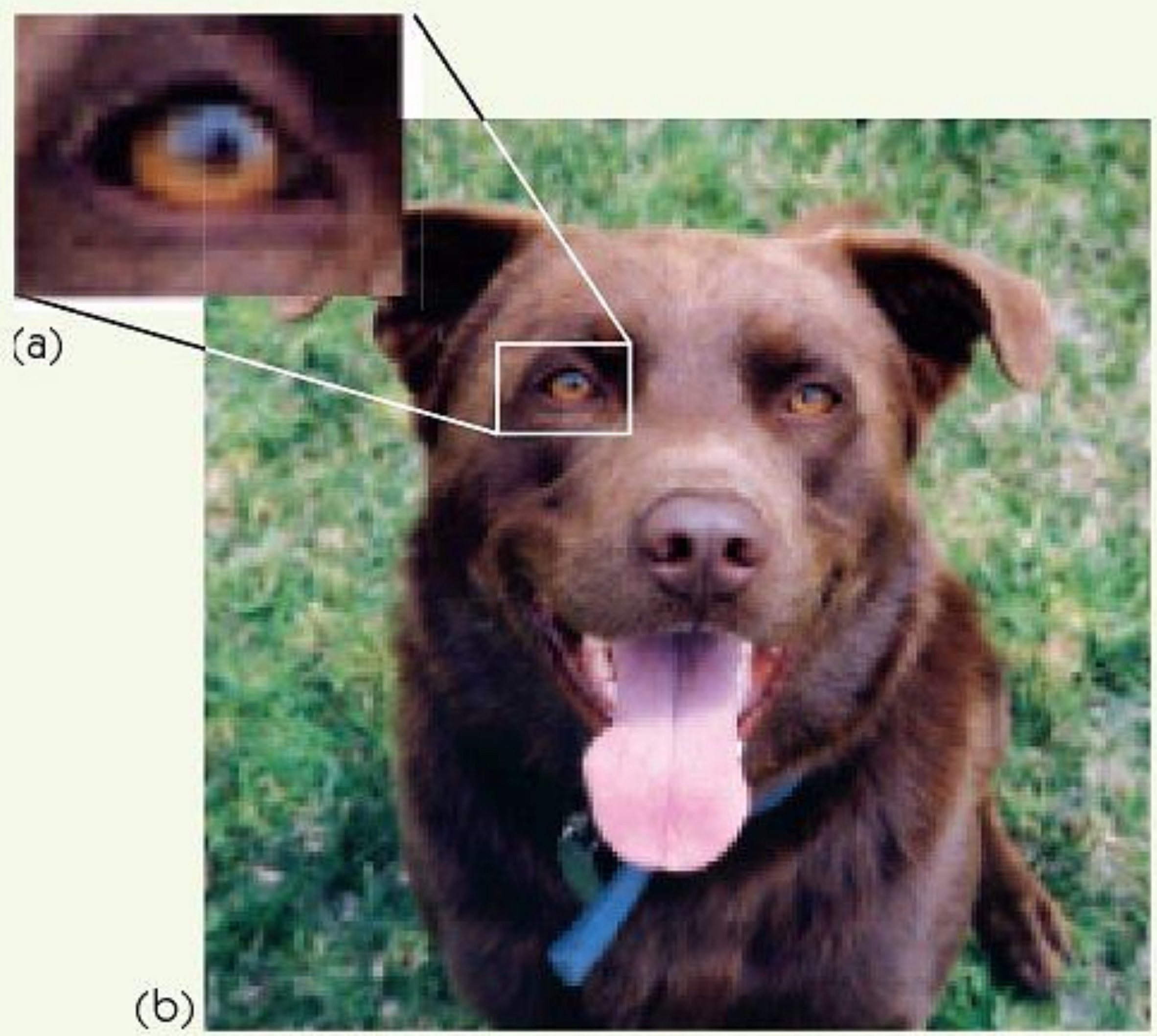
- Storing all of the sampling data for a piece of music requires a very large amount of memory
- We can compress the data in a similar way to the methods we have already seen (lossless)
- Luckily if we don't mind losing some of the data we can use techniques which compress way more (lossy)
- In the case of audio we design our algorithms to throw away data which humans can't perceive
- https://www.youtube.com/watch?v=KGZ0een8vSE (1:20)

IMAGES

- Image sampling: record colour or intensity at fixed, discrete intervals in two dimensions
- Pixels: individual recorded samples
- **RGB encoding scheme**:
  - Colours are combinations of red, green, and blue
  - One byte each for red, green, and blue
- **Raster graphics** store picture as two dimensional grid of pixel values

DIGITIZED IMAGE

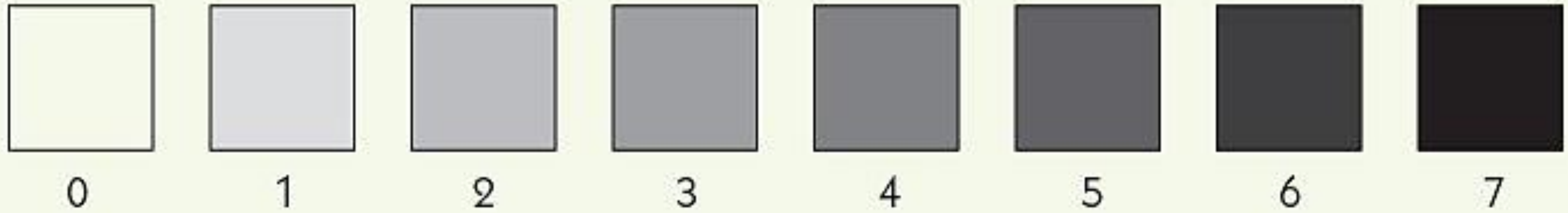
FIGURE 4.7



Example of a digitized photograph
(a) Individual pixels in the photograph
(b) Photograph

USE MORE BITS FOR MORE COLOURS

FIGURE 4.8



An eight-level gray scale

IMAGES AND MOVIES

- We can lose even more information in images and we don't notice it
- Down sample colour
  - https://www.youtube.com/watch?v=n_uNPbdenRs 5:00 minutes in
  - if you are interested he has a really good explanation of the discrete cosine transform used in the jpeg images (this is not part of this course, but it is really interesting) https://www.youtube.com/watch?v=Q2aEzeMDHMA

HOW MUCH DATA?

- What is the space necessary to store the following data? Uncompressed.
    - 1000 integer values - depends on the size of the integer
    - 10-page text paper - depends on how many characters on a page
    - 60-second sound file - depends on sampling rate and bit-depth
    - 480 by 640 image - depends on the colour depth
- **Data compression**: storing data in a reduced-size form to save space/time
    - Lossless: data can be perfectly restored - Huffman, RLE
    - Lossy: data cannot be perfectly restored - most audio and video file types, e.g. mp3, jpeg